

1) الف) درست ✓، $\max(DP[i-1], (P[i] + DP[i-1] - \text{حاصل از } DP[i-1]))$ پس هر خانه‌ی جدول حاصل حدا در آن است.

بسم در دست که حول در سندان کوبیده یعنی حرکت - در دست - یعنی افرایش حد فزونی ترکد روان
اجسام در کیف است پس طبیعتاً اجسام گری از قبل در آن قرار نمی گیرند. چون در هر مرحله
کوبیده یعنی بزرگ می شود. حتماً قدری که در کوبیده یعنی گری مکان ترکد دارد و آن نیز
مکان ترکد در پس مطلقاً از پیم است. حتماً در کوبیده نمی شود!

$$W = 6$$

(2)

$$U = \{25, 20, 15, 40, 50\}$$

$$W = \{3, 2, 1, 4, 5\}$$

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	25	25	25	25
2	0	0	20	25	25	45	45
3	0	15	20	35	40	45	60
4	0	05	20	35	40	60	60
5	0	15	20	35	40	60	65

3 این سؤددیک سؤددی DP است . یک آردید دودیدی شل T آتیب بیکم

$$T(n, z) = \max(T(n-2, z-w) + w, T(n-1, z))$$

$$\text{if } w > z \rightarrow T(n, z) = T(n-1, z)$$

ز در واقع شل نرین از شل استقلال است پس زده تا K است و نا تیر از صورت n

به معنی اضافه شدن کارت جدید است

	0	1	...	n-1	n	...	K
0	0	0	0	0	0	0	0
1	0						
2	0						
...	0						
i	0						
...	0						
n	0						

جواب

Runtime : $O(nk)$

شماره سؤدد knapsack

در واقع یا دوتا کارت قبل را برمی دارد و یا کارت نا را برنمی دارد
پس می تواند کارت نا را انتخاب کند (در سؤدد یا در سؤدد نا را انتخاب کند)

4

نمایش نادرده نظر بگیریم. اگر زمان هر یک از A را شامل می‌شود
 نمایش نادرده نظر می‌گیریم. حال هر زیر مجموعه‌ای قابل ایجاد از این مجموعه (A) را با 2^n نمایش
 پس 2^n زیر مجموعه تولید می‌کند (نمی‌توانیم). در مجموعی A (مجموعه‌ای شامل هر زیر مجموعه‌ای
 A می‌باشد) هر عضو آن از روی این معیار را جابجایی می‌کنیم.

چون زمان هر یک از A را شامل می‌شود. اگر A را به دو دسته تقسیم می‌کنیم. $|A| = 1$ →
 که زمان A را شامل می‌شود. اگر A را به دو دسته تقسیم می‌کنیم. (از هر یک از A و A را
 امکان است که A را به دو دسته تقسیم می‌کنیم. برای تقسیم A را به دو دسته تقسیم می‌کنیم.
 می‌توانیم. اگر A را به دو دسته تقسیم می‌کنیم.

حال برای اعضاء A یک معیار (A, B) داریم. جابجایی ممکن را جابجایی می‌کنیم. $Binary\ search$
 فرآیند جابجایی A را به دو دسته تقسیم می‌کنیم. (A, B) که A را به دو دسته تقسیم می‌کنیم. (میان A و B)
 A حال معیار A است که برای مجموعه‌ای که معیار A را پیدا می‌کنیم. در واقع A را به دو دسته تقسیم می‌کنیم. $P(A)$ را پیدا می‌کنیم.
 سپس برای A و معیار A با فرض اینکه در هر یک از A و B یک معیار A را پیدا می‌کنیم. از A و B را به دو دسته تقسیم می‌کنیم.
 حاصل می‌شود A و B را به دو دسته تقسیم می‌کنیم. از A و B را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم.
 A که A را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم.
 وقت می‌کشید مشخص می‌کنیم. همچنین A را برای A و B را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم.
 مرتبط (مجموعه‌ای که A را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم.)
 می‌توانیم. (مجموعه‌ای که A را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم. A و B را به دو دسته تقسیم می‌کنیم.)

در مجموعه A و B را به دو دسته تقسیم می‌کنیم. و برای هر زیر مجموعه A و B را به دو دسته تقسیم می‌کنیم. و در هر
 مرحله باید روی A و B را به دو دسته تقسیم می‌کنیم. $Binary\ search$ $(\log n)$

پس، $Runtime = O(2^n \times n \times \log(\max(c_i)))$

h

(5)

مسئله یک مسئله DP از نوع knapsack است. یک ملکب یا آریه سه معی DP زنی می‌کشم.
 در یک بعد n حالت نهایی مختلف یک بعد زیبایی‌های موجودند n تا K و رنگ‌های اولیه را به بعد هم
 قرار می‌دهیم. سپس برای $DP(n, 0, k)$ را به مساوی بی‌کتابت قرار می‌دهیم (مانند m_1 باشد) و همچنین

$DP(0, j, k)$ را به ازای هر j و k مساوی 0 می‌گذاریم. $DP = (n+1)(k+1)(m)$

همچنین حالتی $DP(0, 0, k)$ را به ازای هر k مساوی 0 می‌گذاریم.
 مسئله زیبایی با درخت $n-1$
 است و به $n-1$ تا n است
 با یک k رنگ شده است

حالتی رنگی $DP(n, j, k) = \min \left(\begin{matrix} DP(n-1, j-1, k') & \textcircled{1} \\ DP(n-1, j, k) & \textcircled{2} \end{matrix} \right)$
 برای k اعداد گرفته شده

$color\ i \neq color\ i-1$

$color\ i = color\ i-1$

حالتی غیر رنگی $DP(n, j, k) = \min \left(\begin{matrix} DP(n-1, j-1, k') + p(i, k') & \textcircled{1} \\ DP(i-1, j, k) + p(i, k) & \textcircled{2} \end{matrix} \right)$
 برای k اعداد گرفته شده

$color\ i \neq color\ i-1$

$color\ i = color\ i-1$

در هر قسمت \min و $\textcircled{1}$ و $\textcircled{2}$ محاسبه می‌شود و معیار $DP(n, j, k)$ قرار داده می‌شود.

در نهایت طبق دستورالعمل فوق جدول را می‌کشیم و در نهایت اگر هدف آخر رنگ نیست جواب در حالت k از n تا k در نهایت طبق دستورالعمل فوق جدول را می‌کشیم و در نهایت اگر هدف آخر رنگ نیست جواب در حالت k از n تا k در نهایت طبق دستورالعمل فوق جدول را می‌کشیم و در نهایت اگر هدف آخر رنگ نیست جواب در حالت k از n تا k

Runtime: $\textcircled{1} \mathcal{O}(n \times j \times k \times \frac{0+k}{2}) = \mathcal{O}(n \times j \times k^2)$
 با یک \min برای هر k ها

6) چون در بدترین حالت $|E| = |V| + 8$ پس $O(E) = O(V)$ که حدود $O(n)$ هستند

همچنین حدود انجام یافتن دور در گراف نیز اند انجام یکبار BFS است که در $O(E+V)$ است که چون V و E

در گراف خوب از یک دور هستند پس این الگوریتم حاشیه از $O(n)$ انجام میگیرد است.

حال چون $\max(|E|, |V|) = 8$ است پس این گراف حداکثر 9 دور دارد

یک copy در گراف اولیه (یکبار انجام) حال با انجام عبارت BFS و یافتن دور یک دور در این گراف می

باشد پیدا می کنیم. و دور بعدی را با بهترین حدس اول با گراف حذف می کنیم. و مجدداً این کار را

انجام می دهیم تا آنکه $|E'|$ با تعداد ارس آن ~~تعداد ارس آن~~ متفاوت باشد. (یعنی حذف از رسم)

ما به تعداد ارس در تعداد دورهای موجود اول با بهترین حدس را در یک دور میزنیم (حالت یک MST

در تمام حوال عبارت BFS به تعداد Constant اجرا انجام میگیرد پس

Runtime: $O(n)$

یک آرایه شامل وزن های مرغ ها و مرتب شده بر اساس وزن مرغ ها به همراه یک آرایه برای اندازه

تعداد موجود از هر مرغ با وزن معین ایجاد می کنیم. (مرغ ها را دسته بندی می کنیم طوری که هم وزن های دسته بندی شده)

این کار از $O(n \log n)$ است (sort) سپس از کمترین وزن موجود شروع می کنیم.

در هر وزن که می بینیم ۳ حالت داریم: ۱) یک مرغ از آن وزن داشته باشیم. \rightarrow no operation

۲) دو مرغ داشته باشیم \rightarrow از وزن یکی از آنها یکی کم می کنیم

۳) بیش از دو مرغ داشته باشیم \rightarrow از وزن یکی، یک و دهیم و دیگری یک و دهیم

همچنین بر هر دسته (وزن) که می بینیم w در صفا که جمع مرغی از وزن $w-1$ نداشته باشیم در هر صورت

یکی از وزن مرغ موجود را کم می کنیم. در واقع به جردسته ای که می بینیم یک مرغ می دهیم و آن

از وزن $w-1$ نداشته باشیم یکی از وزن آن کم می کنیم سپس به مرغ های وزن w می دهیم و آن جمع مرغی مانده

بود به وزن بعدی می دهیم. آنکه یک مرغ بود نیز مجدداً به دسته بعدی می دهیم. آنکه دو یا بیشتر مانده بود یکی را به خارجی

بعدی بدهد و قدش را یکی بزرگتر کنیم و تفرقه در محل قبلی می ماند (همچنین باید وقت کنیم تغییر وزن تنها برای مرغ های است)

تا این لحظه تغییر وزن نداشته اند اما اگر تعداد اعضای w و تفرقه با $w-1$ اعضای جدید است)

این تفرقه را تا آنجا بایم می گیم. $O(n)$

با یک پیکربندی ساده (دو مرتبه) آرایه تعداد انواع وزن ها را می سازیم $O(n)$

Runtime: $O(n \log n)$

(8) برای حل مسئله دو متغیر داریم که ابتدا هر دو به تعداد صفر مقداردهی شده اند (B, A)

حال A را به $index$ اولین درز و B را به $index$ اولین پلیس $update$ می کنیم.

یک متغیر به نام $counter = 0$ داریم که نشان می دهد تعداد شکست های ما را در یک فرایه می کنیم. حال $|A - B|$

را محاسبه می کنیم اگر از k کمتر بود طبق انتخاب هر چه بهتر درز را، شکست می دهیم و A را به درز بعدی و B را به پلیس

بعدی می بریم. اگر $|A - B| > k$ پس متغیر کوپرت را انتخاب کرده و به منفی مقدار قبلی می بریم.

مثلاً اگر $A < B$ بود A را به درز بعدی می بریم و اگر $A > B$ بود B را به پلیس بعدی می بریم.

و می بیند $|A - B|$ را صاف می کنیم. با چهار در شکست می دهیم $counter + 1 =$ می شود. در نهایت جواب

A یکبار روی کل آرایه و B یکبار روی کل آرایه حرکت کرده اند پس

$$Runtime = O(2n) \text{ و } O(n)$$

9

کلاس ها را بر اساس زمان شروع آنها sort می کنیم $O(n \log n)$
انتخاب می کنیم، در هر مرحله کلاسی را زودتر شروع می شود (کلاس ها) را در یک کلاس بزرگتر می کنیم.

این حله حال می نامی Interval partitioning است.

کلاس اول را انتخاب می کنیم. (زودترین شروع) سپس کلاس دوم را زمان شروع کلاس دوم را به زمان پایان کلاس اول مقایسه می کنیم. اگر کمتر بود یعنی شماره های کلاس ها را یکی قرار می دهیم. اگر بیشتر بود یعنی کلاس دوم را به کلاس اول اضافه می کنیم.



در واقع با یک متغیر مثل n در کلاس ها می بینیم که (از n) و زمان پایان کلاس n را در نظر می گیریم (Finish) سپس یک متغیر n را در کلاس ها می بینیم که (از n) و زمان پایان کلاس n را در نظر می گیریم (Finish) و این کار را تا زمانی که تمام کلاس ها را بررسی کرده ایم.


if ($j - i + 1 > \text{num_class}$)
 $\text{num_class} = j - i + 1$

همین کار را با هر کلاس دیگر می کنیم. تا زمانی که تمام کلاس ها را بررسی کرده ایم.

حالا هم می بینیم که $O(n \log n)$ است و هم $O(n \log n)$ است که این $O(n \log n)$ است.

Run time: $O(n \log n)$

از  اول شروع می‌کنیم (نزدیکترین نقطه 0) و 8 متر بعد از آن (در بست  هم) هر خانه آمد در فاصله 8 متر از 8 متری دکل قرار داشت. از آن عبور می‌کنیم در غیر این صورت در 8 متر بعد از آن یک دکل اضافه می‌کنیم.
حل به روش هزینه‌هاست پس باید ثابت کنیم.

- جهان خلف، فرض می‌کنیم جواب هزینه‌ها به صورت $ALG = \langle P_1, P_2, \dots, P_n \rangle$ باشد و جواب هزینه‌ها به صورت $OPT = \langle q_1, \dots, q_n \rangle$ باشد. در خانه نام باشد، یعنی این دکل جواب است پس در فاصله 8 متر از 8 متری قرار داشت در فاصله 8 متر از 8 متری قرار داشت. قبل از دکل داریم حال یا این خانه دکل دارد پس در این حالت با حذف دکل تا همین‌جا همه خانه‌ها آنتن دارند که تناقض است و OPT هزینه نیست و یا با حذف آن و قرار دادن آن در  فاصله 8 متری خانه همین‌جا دکل است (دکل در OPT در فاصله 8 متر از 8 متری قرار داشت) که یعنی یک مرکز دکل حل هزینه‌ها نزدیک‌تر است که با کنار آن به حل هزینه‌ها می‌رسیم!

Runtime: $O(n)$

حالا باید n خانه را ببینیم!

فرض می‌کنیم این n سکه به صورت صعودی مرتب (sort) شده‌اند

از سکه با کمترین ارزش که به دست می‌آید شروع می‌کنیم و آن را به سکه با بیشترین ارزش اضافه می‌کنیم.

از این به بعد به این ترتیب عمل می‌کنیم. ~~سپس روی سکه‌ها به این ترتیب عمل می‌کنیم.~~ در هر مرحله فرض می‌کنیم

روی سکه‌های با توانایی بیشتر (بزرگتر) و مقدار k را به عنوان ماژیم مقدار سکه‌ها با سکه‌های

موجود تا این لحظه می‌توان از آن‌ها استفاده کرد و در نظر داریم (مثلاً به سکه 1) دو حالت وجود دارد

- 1) اگر از این سکه‌ها فقط از سکه $k+1$ بیشتر و سکه k کمتر به دست می‌آید (از این $k+1$ - مجموعه سکه‌ها)
- 2) اضافه می‌کنیم و مقدار k را به مقدار $2k+1$ تبدیل می‌کنیم اما همچنان روی سکه فعلی هستیم و تفاوت را ادغام می‌کنیم.

در اینجا به سکه‌های بعدی می‌پردازیم.

while $k \leq n$ را تا زمانی که k از n بیشتر نشود (یا اینکه سکه‌های بیشتر به دست می‌آید) ادامه می‌دهیم. در صورتی که k از n بیشتر نشود (تفاوت بیشتر می‌شود) مقدار k را به $2k+1$ تغییر می‌دهیم (تفاوت بیشتر می‌شود) اما اگر به روشی غیر از این مقدار k را به $2k+1$ تغییر می‌دهیم (تفاوت بیشتر می‌شود) این کار را تا زمانی که $k \leq n$ ادامه می‌دهیم.

Runtime: $O(\log K)$

در هر مرحله k را به $2k+1$ تغییر می‌دهیم
تا وقتی که $k \leq n$ باشد
در کارهای دیگر

برهان ملل 1
حج اول: فرض می‌کنیم که به دست می‌آید. راه حل خود را با ALG و راه حل دیگر را OPT می‌نامیم. (اولین تفاوت بین ALG و OPT را در اندیس i نام می‌گذاریم. $ALG[i] < OPT[i]$ در این حالت ما تعداد $ALG[i]$ سکه‌ها را با $OPT[i]$ سکه‌ها مقایسه می‌کنیم. $ALG[i] > OPT[i]$ که اگر $OPT[i]$ کمتر باشد $ALG[i]$ بیشتر است. $ALG[i] = OPT[i]$ که در این حالت ALG و OPT در این مرحله یکسان هستند. $ALG[i] < OPT[i]$ که در این حالت ALG در این مرحله از OPT کمتر است. $ALG[i] > OPT[i]$ که در این حالت ALG در این مرحله از OPT بیشتر است. $ALG[i] = OPT[i]$ که در این حالت ALG و OPT در این مرحله یکسان هستند. $ALG[i] < OPT[i]$ که در این حالت ALG در این مرحله از OPT کمتر است. $ALG[i] > OPT[i]$ که در این حالت ALG در این مرحله از OPT بیشتر است. $ALG[i] = OPT[i]$ که در این حالت ALG و OPT در این مرحله یکسان هستند.